aConCorde: Towards an open-source, extendable concordancer for Arabic

Andrew Roberts, Latifa Al-Sulaiti
and Eric Atwell[1]

Abstract

There is, currently, a surge of activity surrounding Arabic corpus linguistics. As the number of available Arabic corpora continues to grow, there is an increasing need for robust tools that can process this data, whether for research or teaching. One such tool that is useful for both of these purposes is the concordancer – a simple tool for displaying a specified target word in its context. However, obtaining one that can reliably cope with the Arabic language had proved difficult. Also, there was a desire to add some novel features to the standard concordancer to enhance its usefulness within the classroom – easy-to-use root- and stem-based concordance and integration to corpus clustering algorithms are two examples. Therefore, *aConCorde* was created to provide such a tool to the community.

1. Introduction

> 'If a corpus is to be useful, we obviously need to be able to search it quickly and automatically to find examples of a particular linguistic phenomenon (say, a word), to sort the set of examples as required, and to present the resulting list to the user. The kind of program which performs these tasks is a concordancer, and the output it produces is known as a concordance. A concordance is a list of all the examples of the target item (the linguistic phenomenon being searched for), normally accompanied by enough context to enable a human being to study the item's occurrence in detail.'
>
> (Leech and Fligelstone, 1992)

In effect, a concordancer is a tool for summarising the contents of corpora based on words of interest to the user. Lexicographers can use the evidence of a concordance to establish whether a word has multiple senses, and also to help define its meaning (Sinclair, 1987; Zernik, 1991). Concordance tools can be beneficial for data-driven language learning (Johns, 1990).

A number of studies have demonstrated that providing access to corpora and concordancers benefits students learning a second language. Just as lexicographers and linguists can find insights into grammatical structure by studying concordance output, so can language learners (Dodd, 1997). Cobb *et al.* (2001) detailed the improved rates of vocabulary acquisition when using a software tool of which a concordancer was a major component.

[1] *Correspondence to*: Andrew Roberts, *e-mail*: andyr@comp.leeds.ac.uk
address: School of Computing, University of Leeds, Leeds, LS2 9JT, United Kingdom

The simplicity and usefulness of concordancers is such that they should be a valuable tool for any linguist. Unfortunately, as is the general trend within corpus linguistics, research priorities have focused on European languages (although there has been a recent shift towards developing tools in non-European languages such as Chinese, Japanese and Korean). This has meant that there are fewer tools for other languages such as Arabic. When this project began, to the best of our knowledge, there were no readily available concordancers that function correctly with the Arabic language. Arabic is an international language rivalling English in the number of mother-tongue speakers (al-Sulaiti and Atwell, 2006). Graddol (1997) predicts that the number of Arabic mother-tongue speakers will rise from 200 million in 1995 to about 480 million by 2050, whereas English mother-tongue speakers will rise from 360 million to about 500 million within the same period. Considering the growth of interest in Arabic, we felt that Arabic linguists deserved a useful, usable concordancer tool. To this end, the *aConCorde* project was established.

2. Why is Arabic Concordancing Hard?

It is fair to say that for non-Arabic developers, writing tools to process Arabic is more difficult. This perhaps explains why many tools developed for Western languages do not extend well to Arabic. The two most commonly perceived difficulties with Arabic language processing are:

1. The cursive nature of Arabic script means that letters are represented differently depending on whether they occur at the beginning, middle or end of a word. Characters within words are always joined, and never written individually. Vowels are "second-class" characters, in that they can be omitted, and are left for the human reader to infer from context.
2. Arabic is written from right-to-left, as opposed to the majority of other languages that are, of course, written left-to-right.

The process of transliteration is a common approach to resolving these issues. The Buckwalter system (amongst others) will convert Arabic script to an equivalent based on the Roman alphabet (Buckwalter, 2002). It means that the text orientation is also converted, too: the transliterated text reads left-to-right. However, this alone does not fill in missing vowels from the source text. From a computational perspective, transliteration has historically been a necessary step: computers used to be restricted to ASCII (or similar) character sets (i.e., Roman alphabets). However, thanks to the popularity of the Unicode standards since the early 1990s, most modern operating systems and products now support multi-lingual text encodings, and the fonts to display Unicode characters are increasingly common. Implementation of bi-directional text components are also part of most modern operating systems and platform-independent programming languages like Java.

Due to these developments, there is little reason why existing concordance tools have not been adapted to take advantage of the above technologies in order to render the *aConCorde* project redundant even before it had begun. One implementation that sought to provide multi-lingual concordancing, was *xconcord* (Ogden and Bernick, 1996; Boualem *et al*., 1999), developed at the Computing Research Laboratory (CRL), New Mexico University. The CRL produced their own graphical components that were able to display Unicode text and they worked successfully at displaying Arabic text. Even though *xconcord* was ahead of its time when it was first developed in 1996, it did not become a mainstream application. It would still be a useful tool today, but it is hindered because it was written for the Sun Solaris platform. This is not widely used – especially by the average linguist – since this type of system is typically used on expensive high-powered workstations and servers. Unfortunately, therefore, it was not possible to experiment with it as resources required were not available. Development ceased many years ago, but the program can be downloaded[2].

The real difficulty in Arabic concordancing lies not in the displaying of results, but in the inadequacy of searching for Arabic word stems. Commonly, when performing concordance searches, the user wants to see the usage of words all from the same stem. Concordancers tend to offer *wildcard* searches, for example, *perform** (where the asterisk means any character zero or more times) would match *perform*, *performs*, *performer*, *performers*, *performing*, etc. However, Arabic morphology is substantially more complex than English. Arabic words are derived from a root of typically three consonants, and each root has a set of patterns which can add additional characters as an affix, a prefix or even an infix. An example would be the transliterated Arabic root *ktb* (write). Patterns associated with that root can produce words semantically linked, including verbs like *kataba* (he wrote) and *naktubu* (we write), and many nouns like *kitAb* (book), *maktuub* (letter) and *maktaba* (library)[3]. However, performing a wildcard search like **k*t*b** would return too many matches, many of which are not associated with the *ktb* root, simply because they happen to contain these consonants.

With the exception of the Qur'an, the lack of vowels in modern written text provides difficulties too. This leaves a great deal of ambiguity that is only resolved when looking at the context of a given word. An English example may be *fr* that could be *for*, *fir*, *fur*, *far*, *four*, *fear*, *fair*, *fire*, *afar*, *afore*, etc. Yet, if you can only search for *fr* even though you are only interested in *far*, it would clearly be frustrating to have to read through irrelevant results. It is unlikely that these specific issues will be resolved by generic concordancers.

3. Arabic Concordance with *MonoConc*, *Wordsmith*, *Xaira* and *aConCorde*

For the task of concordance, the market leaders are *MonoConc* (Lawler, 2000) and *WordSmith* (Scott, 2004). Both are commercial products and are well-established tools for text analysis. They are only available on the Microsoft Windows platform. *Xaira* (Burnard, 2004) is a relative newcomer but stems from the well-known *SARA* toolkit for users of the British National Corpus. This section primarily seeks to compare the ability of retrieving concordance output from Arabic corpora using these established tools.

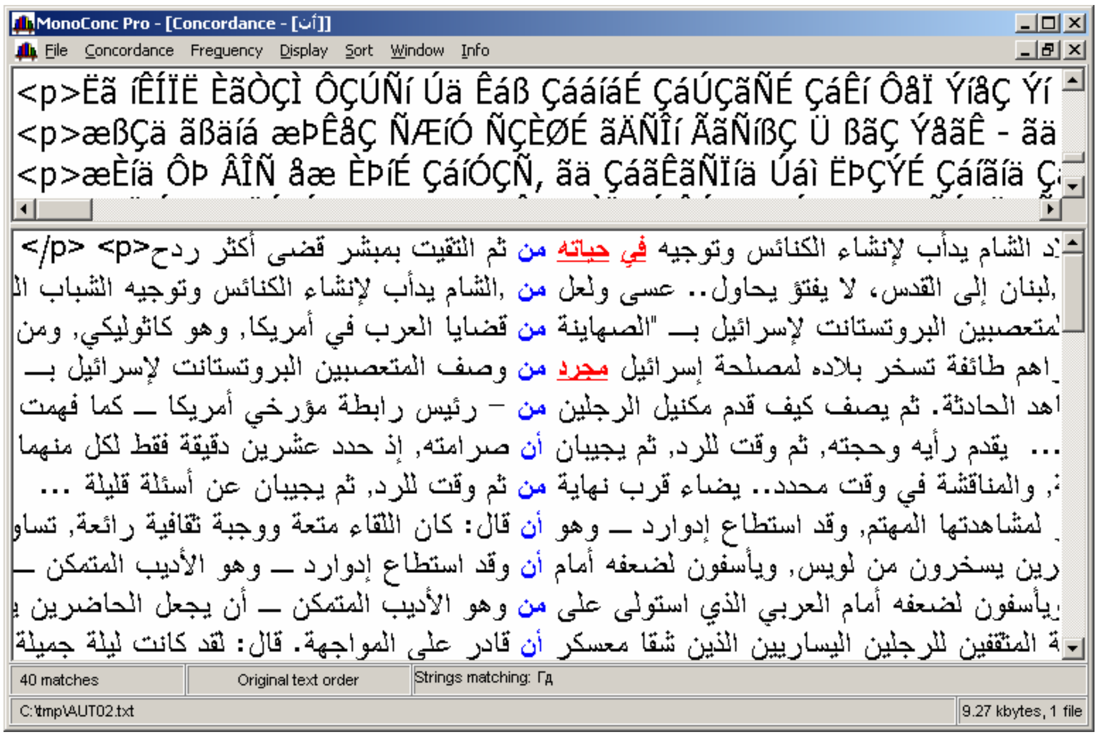
The corpus used in the comparisons is the Corpus of Contemporary Arabic, or CCA (al-Sulati, 2004), consisting of 850,000 words of various text-types in over 400 files. This corpus is annotated according to the TEI standards using XML markup, and all files are encoded in the 8-bit Unicode standard, UTF-8. A key aim of the CCA was that it should be for use within a pedagogical context to supplement the teaching of Arabic as a foreign language using corpus-based approaches.

[2] http://crl.nmsu.edu/software/

[3] Many sources cover Arabic grammar, however, issues about Arabic and computing are well introduced by Khoja (2003) and de Roeck (2002).

3.1 *MonoConc*

Developed by a linguist, Michael Barlow, for language teachers and researchers, *MonoConc* has proved to be very popular for language analysis because it comes with many additional features, such as collocation discovery and tag-sensitive searches. The *MonoConc* website[4] states, 'Some users have been using *MonoConc/Pro* with Arabic. I don't have any details of their procedures.' Unfortunately, despite initial optimism, using Arabic texts with *MonoConc* was not totally successful. Issues that arose during experiments were as follows:



```
<p>Ëã íÊÍÏË ÈãÒÇÌ ÔÇÚÑí Úä Êáß ÇáÃíÇã ÇáÚÇãÑÉ ÇáÊí ÔåÏ ÝíåÇ Ýí
<p>æßÇä ãßäíá æÞÊåÇ ÑÆíÓ ÑÇÈØÉ ãÄÑÎí ÃãÑíßÇ Ü ßãÇ Ýåãã - ãä
<p>æÈíä ÔÞ ÂÎÑ åæ ÈÞíÉ ÇáíÓÇÑ, ãä ÇáãÊãÑÏíä Úáì ËÞÇÝÉ ÇáíãíäÇ
```

```
د الشام يدأب لإنشاء الكنائس وتوجيه في حياته من ثم التقيت بمبشر قضى أكثر ردح<p> </p>
لبنان إلى القدس، لا يفتؤ يحاول.. عسى ولعل من الشام يدأب لإنشاء الكنائس وتوجيه الشباب ال
متعصبين البروتستانت لإسرائيل بـــ "الصهاينة من قضايا العرب في أمريكا, وهو كاثوليكي, ومن
اهم طائفة تسخر بلاده لمصلحة إسرائيل مجرد من وصف المتعصبين البروتستانت لإسرائيل بـــ
اهد الحادثة. ثم يصف كيف قدم مكنيل الرجلين من – رئيس رابطة مؤرخي أمريكا ـــ كما فهمت
... يقدم رأيه وحجته, ثم وقت للرد, ثم يجيبان أن صرامته, إذ حدد عشرين دقيقة فقط لكل منهما
, والمناقشة في وقت محدد.. يضاء قرب نهاية من ثم وقت للرد, ثم يجيبان عن أسئلة قليلة ...
لمشاهدتها المهتم, وقد استطاع إدوارد ـــ وهو أن قال: كان اللقاء متعة ووجبة ثقافية رائعة, تساو
رين يسخرون من لويس, ويأسفون لضعفه أمام أن وقد استطاع إدوارد ـــ وهو الأديب المتمكن ـــ
يأسفون لضعفه أمام العربي الذي استولى على من وهو الأديب المتمكن ـــ أن يجعل الحاضرين ب
ة المثقفين للرجلين اليساريين الذين شقا معسكر أن قادر على المواجهة. قال: لقد كانت ليلة جميلة
```

```
40 matches    Original text order    Strings matching: Гд
C:\tmp\AUT02.txt    9.27 kbytes, 1 file
```

Figure 1: Arabic concordance using *MonoConc*[5]

Unicode support. It appeared that *MonoConc* does not support texts encoded in UTF-8. It was therefore necessary to convert texts to the Windows Arabic Codepage-1256 before the program would display the Arabic text correctly.

Incorrect concordance order. Perhaps the most significant problem for performing Arabic concordance is that the output is in the wrong order (See Figure 1). *MonoConc* does not take the right-to-left nature of Arabic into account when displaying results. For an English reader, it would be equivalent to seeing the output: "on the mat. [sat] The cat", rather than "The cat [sat] on the mat." (where the bracketed word is the target word). This was observed by Hoogland (2003) during the Nijmegen Dutch-Arabic Dictionary Project, 'This seemed a serious shortcoming in the beginning, but soon we experienced that we got used to this very quickly.' Hoogland *had* to get used to it because it was better to have a broken concordance

[4] http://www.monoconc.com

[5] Discovered collocates are underlined, although they should not be there. For unknown reasons the text in the top pane did not display using Arabic fonts.

than none at all. It is clearly far from ideal – it would be a problem for teaching purposes in particular as it would confuse the learner. (N.B. When the concordance results are saved to a file, and this file is viewed with a text editor, the ordering issue disappears.)

'Noisy collocates'. A strange artefact appeared in the concordance output whenever a match included a discovered collocate. It was default behaviour for the software to highlight found collocates. However, the only problem was that these collocates were actually inserted into the wrong position within the concordance context, and as a result caused a certain amount of interference. This was remedied by turning off the option to highlight collocates, after which the highlighted words would then vanish from the context.

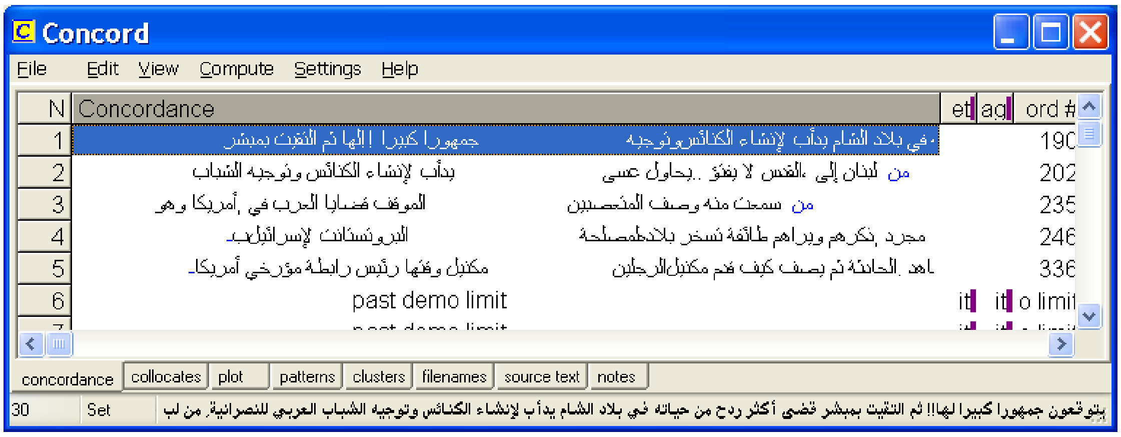
Addition of unwanted terms. Despite a simple search term being submitted to *MonoConc*, sometimes it included matches within the results that are not equal to the target word. This behaviour appears to be random as it only occurs with some words and not others. This is illustrated, too, in Figure 1 where the central column contains tokens that were different from the search term.

3.2 *WordSmith*

WordSmith was first released in 1996 and is still developed by a linguist, Mike Scott. *WordSmith* is currently at version 4 and sports three tools: Wordlist, Keyword and Concord. The final tool is of interest in this report, but it should be obvious what the others do. The documentation notes that *WordSmith* supports Unicode, which obviously lends itself to the analysis of the CCA. A demo version is available, which was used for this report. It retains all the functionality of the full version, but significantly limits the number of results returned when performing queries.

A degree of success was initially achieved when using the tool during September 2005, in that it was possible to get the tool to display Arabic script – which means its Unicode support was working. Unfortunately, like *MonoConc* it displayed the prior and posterior contexts in the opposite order required for a right-to-left language. The matches are displayed in two columns, the left column must be read from right-to-left first, and then the reader can continue with the second column, which begins with the target word, followed by the rest of the context (see Figure 2).

We were only recently made aware that *WordSmith* had in fact been updated to cope better with Arabic concordance. We had initially overlooked this because the version number for the latest issue of *WordSmith* had not been incremented to indicate any change. We are pleased to report that the tool does in fact work properly with Arabic text, using the correct ordering. That said, it did require a little more configuring, such as enabling right-to-left support at the level of the Windows operating system. You also need to spend some time in the *WordSmith* preferences dialogs to add Arabic to its supported language set and then select it to be used within the current session. Other Arabic colleagues who we asked to experiment with *WordSmith* had great difficulty achieving a result. The feeling was that whilst the support was clearly there, it was not necessarily as accessible as they would have liked it.



N	Concordance		ord #
1	في بلاد الشام يدأب لإنشاء الكنائس وتوجيه جمهورا كبيرا لها!! ثم التقيت بمبشر		190
2	من لبنان إلى القدس لا يفتؤ ..يحاول عسى يدأب لإنشاء الكنائس وتوجيه الشباب		202
3	من سمعت منه وصف المتعصبين الموقف قضايا العرب في أمريكا, وهو		235
4	مجرد تكرهم ويراهم طائفة تسخر بلادهالمصلحة البروتستانت لإسرائيل		246
5	اهد الحادثة ثم يصف كيف فهم مكتبل الرجلين مكتبل وقتها رئيس رابطة مؤرخي أمريكا		336
6	past demo limit		it o limit

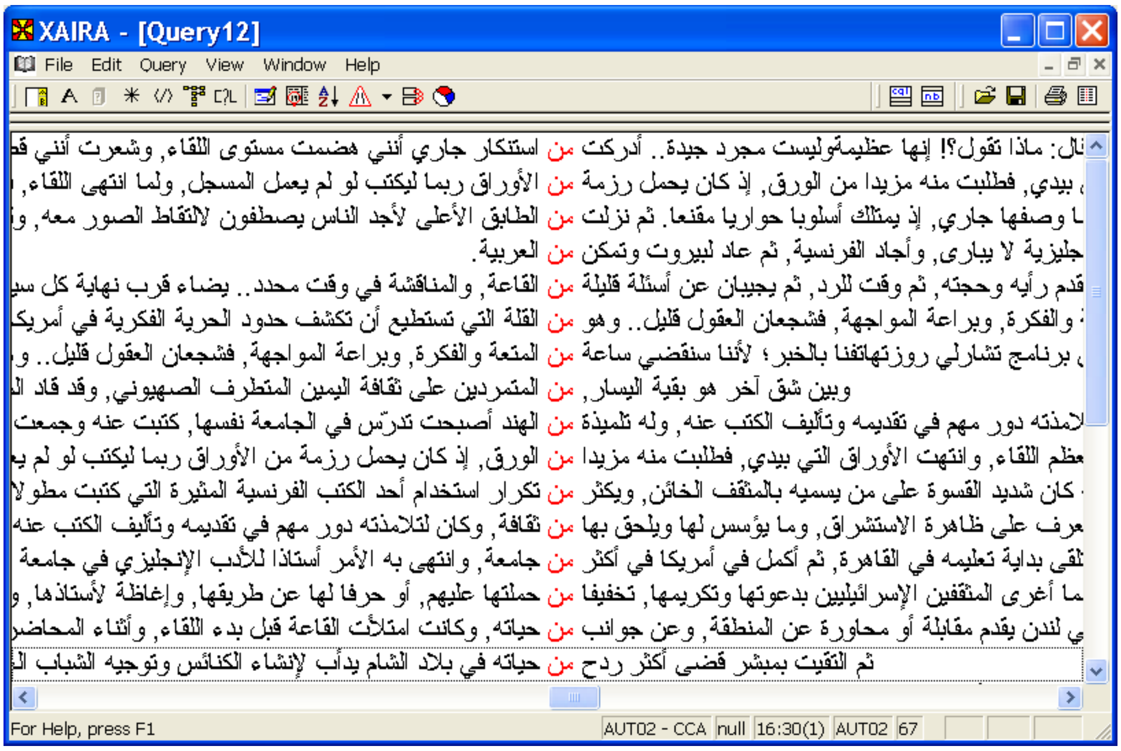
concordance | collocates | plot | patterns | clusters | filenames | source text | notes

30 Set يتوقعون جمهورا كبيرا لها!! ثم التقيت بمبشر قضى أكثر ردح من حياته في بلاد الشام يدأب لإنشاء الكنائس وتوجيه الشباب العربي للنصرانية, من لب

Figure 2: "Incorrect" Arabic concordance as displayed by *WordSmith* (now corrected in the latest version)

3.3 *Xaira*

Xaira is a new tool designed to supersede *SARA* (Dodd, 1997; Aston and Burnard, 1998) – an application for text analysis on the British National Corpus. *Xaira* is no longer tied to the one corpus, instead opting for a more generalised application. It takes advantage of Unicode and XML technologies to achieve this (Burnard and Dodd, 2003). At the time of writing, its first version is still in beta testing. It can perform complex searches as it can filter results according to the XML annotation.



ال: ماذا تقول؟! إنها عظيمةوليست مجرد جيدة.. أدركت من استنكار جاري أنني هضمت مستوى اللقاء, وشعرت أنني قد
بيدي, فطلبت منه مزيدا من الورق, إذ كان يحمل رزمة من الأوراق ربما ليكتب لو لم يعمل المسجل, ولما انتهى اللقاء,
ا وصفها جاري, إذ يمتلك أسلوبا حواريا مقنعا. ثم نزلت من الطابق الأعلى لأجد الناس يصطفون لالتقاط الصور معه, و
جليزية لا يبارى, وأجاد الفرنسية, ثم عاد لبيروت وتمكن من العربية.
قدم رأيه وحجته, ثم وقت للرد, ثم يجيبان عن أسئلة قليلة من القاعة, والمناقشة في وقت محدد.. يضاء قرب نهاية كل سي
والفكرة, وبراعة المواجهة, فشجعان العقول قليل.. وهو من القلة التي تستطيع أن تكشف حدود الحرية الفكرية في أمريك
برنامج تشارلي روزتهاتفنا بالخبر؛ لأننا سنقضي ساعة من المتعة والفكرة, وبراعة المواجهة, فشجعان العقول قليل.. و
وبين شق آخر هو بقية اليسار, من المتمردين على ثقافة اليمين المتطرف الصهيوني, وقد قاد ال
لامذته دور مهم في تقديمه وتأليف الكتب عنه, وله تلميذة من الهند أصبحت تدرّس في الجامعة نفسها, كتبت عنه وجمعت
عظم اللقاء, وانتهت الأوراق التي بيدي, فطلبت منه مزيدا من الورق, إذ كان يحمل رزمة من الأوراق ربما ليكتب لو لم يع
كان شديد القسوة على من يسميه بالمثقف الخائن, ويكثر من تكرار استخدام أحد الكتب الفرنسية المثيرة التي كتبت مطولا
عرف على ظاهرة الاستشراق, وما يؤسس لها ويلحق بها من ثقافة, وكان لتلامذته دور مهم في تقديمه وتأليف الكتب عنه
لقى بداية تعليمه في القاهرة, ثم أكمل في أمريكا في أكثر من جامعة, وانتهى به الأمر أستاذا للأدب الإنجليزي في جامعة
ما أغرى المثقفين الإسرائيليين بدعوتها وتكريمها, تخفيفا من حملتها عليهم, أو حرفا لها عن طريقها, وإغاظة لأستاذها, و
ي لندن يقدم مقابلة أو محاورة عن المنطقة, وعن جوانب من حياته, وكانت امتلأت القاعة قبل بدء اللقاء, وأثناء المحاضر
ثم التقيت بمبشر قضى أكثر ردح من حياته في بلاد الشام يدأب لإنشاء الكنائس وتوجيه الشباب ال

For Help, press F1 | AUT02 - CCA | null | 16:30(1) | AUT02 | 67

Figure 3: Example of *Xaira* producing correct concordance output

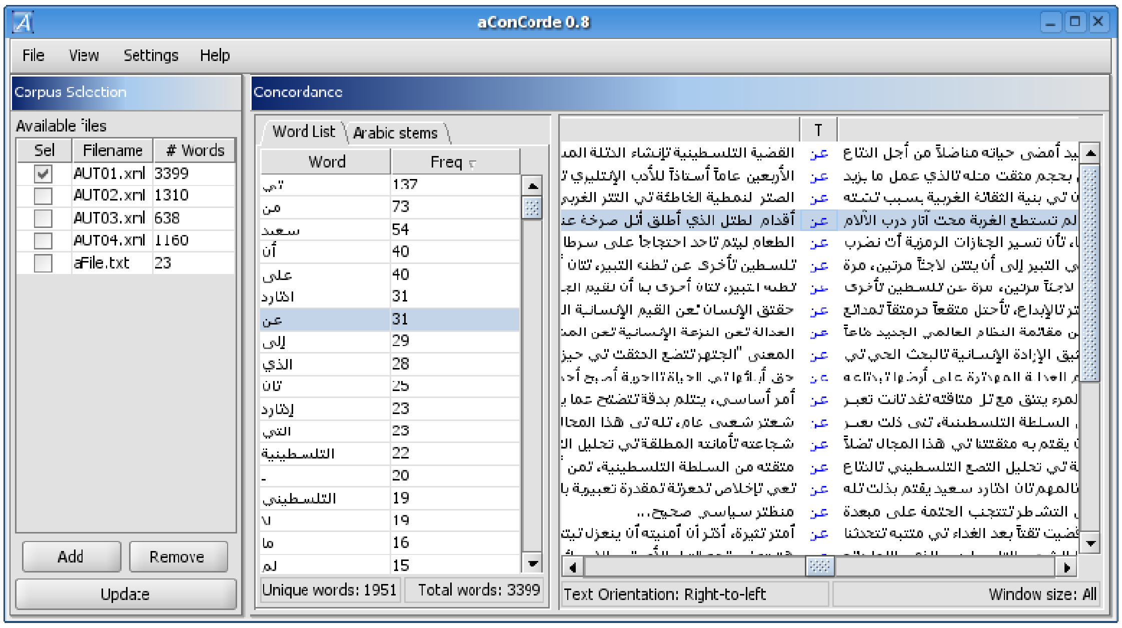
For example, a spoken corpus is commonly annotated with information such as the gender of the current speaker. Therefore, *Xaira* is able to perform searches based on male language usage and compare it to female usage.

Xaira does not suffer any problems in that it displays the entire concordance in the correct order (see Figure 3). However, in order to get to the point of looking at results, the user must go through a relatively long procedure, using an additional tool to prepare the source texts into the format expected by the software. This need not be a problem for dedicated computational linguists, but may deter more casual users, such as linguists or language teachers wanting to use corpus and concordance evidence only occasionally.

3.4 *aConCorde*

aConCorde (Roberts and Atwell, 2005) is neither a commercial product nor a research project. In terms of features, it is relatively basic when compared to the systems already discussed. The design aim was to ensure it was as multi-lingual as possible, with extra emphasis on the Arabic language. It will be no surprise, therefore, that it works well for right-to-left languages (see Figure 4). No additional configuration of both the operating system or the application is required. *aConCorde* cannot detect the language of selected texts, but it can detect easily whether the language is left-to-right or right-to-left. Therefore, it will adjust its concordance display automatically depending on the texts currently being analysed.

An additional feature that is useful for Arabic users is the provision of an Arabic interface. Not only does this provide Arabic translations for all the menus, buttons *etc*., but even switches the entire application layout to right-to-left. In theory it is relatively easy to add additional language support, although it can be difficult to find willing translators.



Sel	Filename	# Words
✓	AUT01.xml	3399
	AUT02.xml	1310
	AUT03.xml	638
	AUT04.xml	1160
	aFile.txt	23

Word	Freq
في	137
من	73
سعيد	54
أن	40
على	40
ادوارد	31
عن	31
إلى	29
الذي	28
كان	25
إدوارد	23
التي	23
الفلسطينية	22
-	20
الفلسطيني	19
لا	19
ما	16
لم	15

Unique words: 1951 Total words: 3399

	T	
القضية الفلسطينية لإنشاء الدولة المد	عن	يد أمضى حياته مناضلاً من أجل الدفاع
الأربعين عاماً أستاذاً للأدب الإنكليزي	عن	بحجم مثقف مثله فالذي عمل ما يزيد
الصور النمطية الخاطئة في الفكر الغربي	عن	في بنية الثقافة الغربية بسبب كتبته
أقدام لطفل الذي أطلق أول صرخة عن	عن	لم تستطع الغربة محو آثار درب الآلام
الطعام ليتم واحد احتجاجاً على سرطا	عن	فأن تسير الجنازات الرمزية أو نضرب
فلسطين وأخرى عن وطنه التعبير، وثان	عن	التعبير إلى أن يكون لاجئاً مرتين، مرة
وطنه التعبير، وثان أخرى بما أن نقيم الع	عن	لاجئاً مرتين، مرة عن فلسطين وأخرى
حقوق الإنسان وعن القيم الإنسانية ال	عن	ر فالإبداع، وأحتل موقعاً مرموقاً كمدافع
العدالة وعن النزعة الإنسانية وعن المد	عن	ن مقاومة النظام العالمي الجديد دفاعاً
المعنى "الجوهر" وتضع الحقيقة في حيز	عن	يق الإرادة الإنسانية فالبحث الحي في
حق أبنائها في الحياة الحرة أصبح أحد	عن	م العدالة المهدورة على أرضها والدفاع
أمر أساسي، يتكلم بدقة وتفصح عما ي	عن	لمرء يتفق مع كل مواقفه فقد كانت تعبر
شعور شعبي عام، كله في هذا المجا	عن	السلطة الفلسطينية، كي ذلك يعبر
شجاعته وأمانته المطلقة في تحليل ال	عن	يقوم به مثقفنا في هذا المجال فضلاً
مثقفه من السلطة الفلسطينية، كمن	عن	ة في تحليل الوضع الفلسطيني فالدفاع
تعبر بإخلاص كمعرفة كمقدرة تعبيرية با	عن	فالمهم كان ادوارد سعيد يقوم بذلك كله
منظور سياسي صحيح...	عن	التشطر وتتجنب الحتمية على مبعدة
أكثر كثيرة، أكثر أن أمنيته أن ينعزل كيت	عن	قضيت وقتاً بعد الغداء في مكتبه نتحدث

Text Orientation: Right-to-left Window size: All

Figure 4: Example of *aConCorde* displaying Arabic concordance correctly

3.5 Discussion

The *aConCorde* project was started solely because we could not find a concordancer to work correctly with Arabic. The *Xaira* software has been demonstrated to work with Arabic too, so it could be argued that there was no need for *aConCorde*. Unfortunately, *Xaira* was only discovered *after* development had already begun. (And the same goes for the recent discovery of *WordSmith*'s recent update.) Initially, it might seem that *Xaira* and *WordSmith* have now rendered *aConCorde* redundant, but there is a market for both. Packages like *Xaira* are very sophisticated, but with increased power and flexibility comes additional complexity. It certainly lends itself to research purposes requiring complex queries for fine-grained language analysis. However, it is less appropriate in the language teaching environment. Teachers and students may be put off by the complex interface and the amount of effort needed to get a concordance output. By contrast, *aConCorde* is relatively simple to get up and running – literally select a corpus to use, then either type in your query or select *word* from the word frequency panel and you will be presented with your results. This approach could well be more suited to this audience. Also, *aConCorde* does provide special searches that are specific to Arabic analysis in the form of root/stem-based concordance, and this is not found in any other package. Finally, the *aConCorde* software is free to use by anyone (as is the source code).

4. The *aConCorde* System

The *aConCorde* project was originally conceived as a prototype to illustrate the application of new developments in software engineering, specifically Java programming language support for language-independent (and direction-independent) processing of Unicode character sets, rapid development of graphical user interfaces, and ease of integration with and into other open-source code and tools. This makes it much easier to write tools to support not just the Arabic language, but practically any language. Within one afternoon, a very simple multi-lingual concordance engine was programmed, and during the following afternoon, a basic graphical user interface was added. The user is able to switch between a native English or Arabic interface. All Arabic text is displayed correctly, all concordance is output as expected by an Arabic reader, and there is no transliteration required for the input or output. This prototype was very crude and would not have been adequate for proper use in the real world – it was simply a quick experiment to prove that the display of Arabic concordance was in fact fairly trivial. Of course, developing a more robust, feature-packed and user-friendly tool is infinitely more difficult and time consuming.

4.1 *aConCorde* features

Subsequent improvements and extensions have allowed *aConCorde* to mature into a more reliable and useful tool. The *aConCorde* system contains a number of features that make it to stand out from competing products, including:

- full Arabic support (no need to transliterate to Roman alphabet before concordance)
- English and Arabic native interfaces

- multi-platform functionality – can run on most major operating systems. It supports an extensive range of character encodings, including Unicode (UTF-16 and UTF-8), Windows Arabic (CP1256), IBM Arabic (CP420), MacArabic, ISO Latin/Arabic (ISO 8859-6) and ASCII encoding
- multi-format support that allows you to load text files, XML files, HTML files, RTF files and MS-Word files directly
- can save results either as a plain text file, or HTML file that can keep the alignment of the concordance
- word frequency analysis
- concordance that can be sorted on left or right contexts
- a range of query options: key-word, phrase, proximity, boolean, wildcard and Arabic root/stem queries, and
- *aConCorde* is freeware and open-source (released under the General Public License[6]).

The *aConCorde* system is built using the Java programming language. Java allows the programmer to produce software for the most popular operating systems without any extra effort, which is why *aConCorde* will run correctly on Microsoft Windows, Linux or Mac OS (and others) providing the user has the Java Runtime Environment installed. It is important to note that many earlier Arabic tools were reliant on Arabic Windows (a localised version of Microsoft Windows with support for Arabic script and right-to-left interfaces). *aConCorde* will of course run on standard Windows or Arabic Windows.

The Java platform is also one of the reasons why producing a multi-lingual capable application was straightforward. Internationalisation features were an important aspect of the design of Java. For instance, Java's internal mechanism to store text uses the Unicode standard, and all visual components have in-built support for left-to-right or right-to-left languages (some components can even cope with Japanese top-to-bottom text orientation).

4.2 Root- and stem-based concordance

As discussed in Section 2 the wildcard searches that make it reasonably simple for stem-based concordance in Western languages do not scale universally. To recap, an Arabic root is typically a sequence of three consonants (although two, four and five consonant roots exist) which forms the basis on which words are constructed. For example, the root *ktb* (write) has stems derived such as *katab* (write) and *iktatab* (register). From the stem you can derive the various morphological forms using affixes, infixes and suffixes, e.g., *Taktub* (she writes), *yaktub* (he writes), *aktub* (I write), and so on.

4.2.1 Buckwalter's morphological analyser

Buckwalter's morphological analyser is distributed by the LDC and has been used by many projects including the LDC's Arabic Treebank project (Maamouri *et al.*, 2004).

[6] http://www.gnu.org/copyleft/gpl.html

It consists of three lexicons (affixes, suffixes and stems) and three sets of rules that specify how these lexicons can be intersected to derive Arabic word tokens. The analyser reads in a file and processes each word in isolation. By utilising the lexicons and rules, it can break down the input token and produce all possible valid morphological solutions. In cases where there is more than one solution, it does not offer any hints as to which is the most probable analysis.

The databases and the analyser work with Buckwalter's transliteration system. Until recently, direct input/output of Arabic characters in computer systems has not been trivial. Buckwalter's system is a one-to-one mapping of the Roman alphabet and the Arabic alphabet. Conveniently, there is also a one-to-one mapping to the Unicode character encoding standard.

There are other transliteration systems in use although there is no single official transliteration standard. Buckwalter's transliteration is not recognised within the Arabic linguistic community at large (Beesley, 2003), since it is relatively modern, and aimed at the computational processing of Arabic. Its use of Latin punctuation symbols for Arabic letters is clearly less intuitive to a human reader than some of the more phonetic-based transliteration systems. For example, Buckwalter uses '$' for Sheen, whereas others, such as Qalam (Heddaya, 1985), use 'sh'.

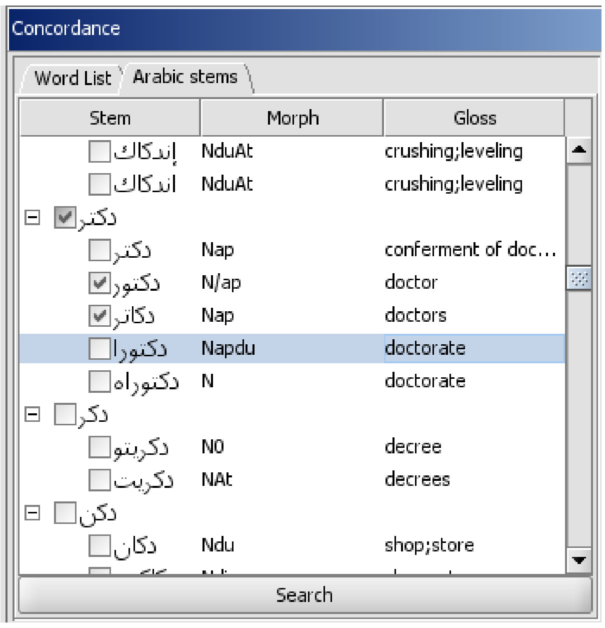
4.2.2 Buckwalter and *aConCorde*

If a user wished to produce a concordance for all words derived from the stem *katab*, a wildcard search approach is, clearly, not feasible. The alternative is to search for a manually hand-crafted list of derived words. However, with *aConCorde*, this burden is removed. Using the databases of Buckwalter's morphological analyser, *aConCorde* can provide the roots and stems *a priori* to the user. They can then select to search for one or more stems, or even everything within a given root. Buckwalter's databases were converted from Buckwalter's transliteration alphabet to Unicode so that the root/stems are displayed in native Arabic (see Figure 5).

There is a limitation here in that the user is always presented with the full database, even if many of the terms are not present within the corpus being currently analysed. When loading an Arabic corpus, it might be preferable to analyse each word and determine its stem and root, and then only have these entries visible. However, real-time stemming of a corpus in Arabic is difficult due to the complex morphology. For example, it is quite easy to remove accidentally those affixes that were in fact part of the word. That said, there are stemmers that exist, such as Khoja's (2003) stemmer used in her APT tagger.

4.3 Similar terms and word clusters

A novel feature that is specific to *aConCorde* is the use of similarity functions. These are borrowed from the information retrieval (IR) domain. They are typically used to find similar documents based on the terms contained within. In *aConCorde* its low-level model, from an IR perspective, regards sentences as "documents". The concordancer has a set of term-frequency vectors and these can be plotted into a vector space. During the concordance, the user can specify their search term, select a sample line from the concordance and its term-vector will be compared to the others using cosine similarity (Jurafsky and Martin, 2000: 650) and then be presented with similar sentences from within the corpus.



Stem	Morph	Gloss
☐ إندكاك	NduAt	crushing;leveling
☐ اندكاك	NduAt	crushing;leveling
⊟ ✓ دكتر		
☐ دكتر	Nap	conferment of doc...
✓ دكتور	N/ap	doctor
✓ دكاتر	Nap	doctors
☐ دكتورا	Napdu	doctorate
☐ دكتوراه	N	doctorate
⊟ ☐ دكر		
☐ دكريتو	N0	decree
☐ دكريت	NAt	decrees
⊟ ☐ دكن		
☐ دكان	Ndu	shop;store

Figure 5: Example of root/stem selection within *aConCorde* using Buckwalter's stem database

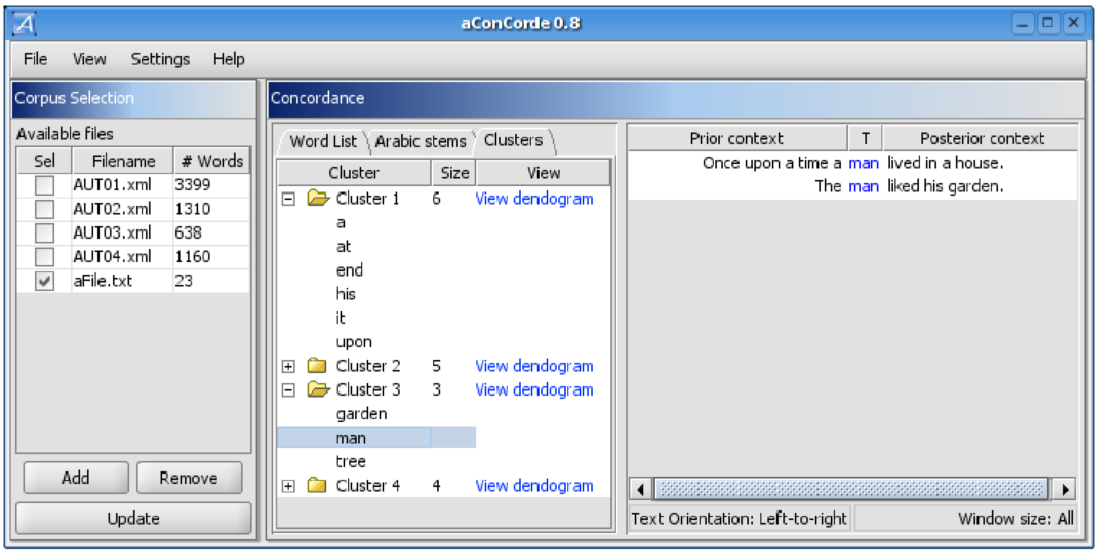
4.3.1 Clustering and *aConCorde*

aConCorde provides a means for the user to interface external clustering code and display the output within the concordancer itself (see Figure 6). Section 5 illustrates the advantage of Java open-source development: more potential features could be readily added to concordance software to enable alternative perspectives and analyses of a corpus.

The term similarity and clustering features were implemented primarily for the use of concordance within the learning environment. Concordance has already proved successful within the classroom, but this concordancer can provide additional information to the user to assist in building vocabularies and grammatical patterns.

5. Clustering

The principle of clustering algorithms is to divide a set of objects into clusters. By using an appropriate measure of similarity, a good clustering algorithm will place objects that are similar into the same cluster, whereas dissimilar ones are clustered into different groups. It has the advantage of being truly unsupervised, i.e., it requires no prior knowledge about the data being clustered. Clustering is a broad subject and has been widely applied. Within computational linguistics, it has been used successfully in syntactic (Atwell and Drakos, 1987; Finch and Chater, 1992; Hughes, 1994) and semantic (Ibrahimov *et al.*, 2001) classification and information retrieval systems.



Sel	Filename	# Words
☐	AUT01.xml	3399
☐	AUT02.xml	1310
☐	AUT03.xml	638
☐	AUT04.xml	1160
✓	aFile.txt	23

Cluster	Size	View
Cluster 1	6	View dendogram
a		
at		
end		
his		
it		
upon		
Cluster 2	5	View dendogram
Cluster 3	3	View dendogram
garden		
man		
tree		
Cluster 4	4	View dendogram

Prior context	T	Posterior context
Once upon a time a	man	lived in a house.
The	man	liked his garden.

Text Orientation: Left-to-right   Window size: All

Figure 6: Cluster view within *aConCorde*

Clustering is a mathematical procedure that merges points in a vector space that are close to each other. To cluster words, a method is required to represent a given word into a vector to be plotted. The approach used here is that of Roberts (2002), which records collocation frequencies of a given content word relative to function words, for a specified window size. This data can be converted simply to a vector ready to be clustered.

5.1 Roberts' Method

Function words have an important role in language. The words themselves contain little meaning but instead specify relationships between other (content) words. Function words tend to be used frequently. The rationale behind Roberts' method is that if function words are so important and frequent, all content words within a corpus should co-occur with them a number of times, even within a small window. An hypothesis follows from this: words that co-occur similarly with the same function words could be considered grammatically similar, in that they share the same word class.

For a given corpus, *T*, you can construct a set of function words, *F* (see Section 5.2), and a set of content words, *C* (i.e., a word list of all the words in the corpus). A window size, *w*, must also be selected[7]. Select a content word, *c*, from *C* and then select a function word, *f*, from *F*. Scan the corpus for every instance of *c*. For each one, check if *f* co-occurs within the window, *w*. If it does, record the relative position from the target content word. Figure 7 illustrates such an example, showing how the content word 'first' co-occurs with the function word 'the' in a window size of four.

[7] In this paper, a window size of *n* means *n* words before and *n* words after a target word, thus spanning 2*n*+1 words in total, including the target word.

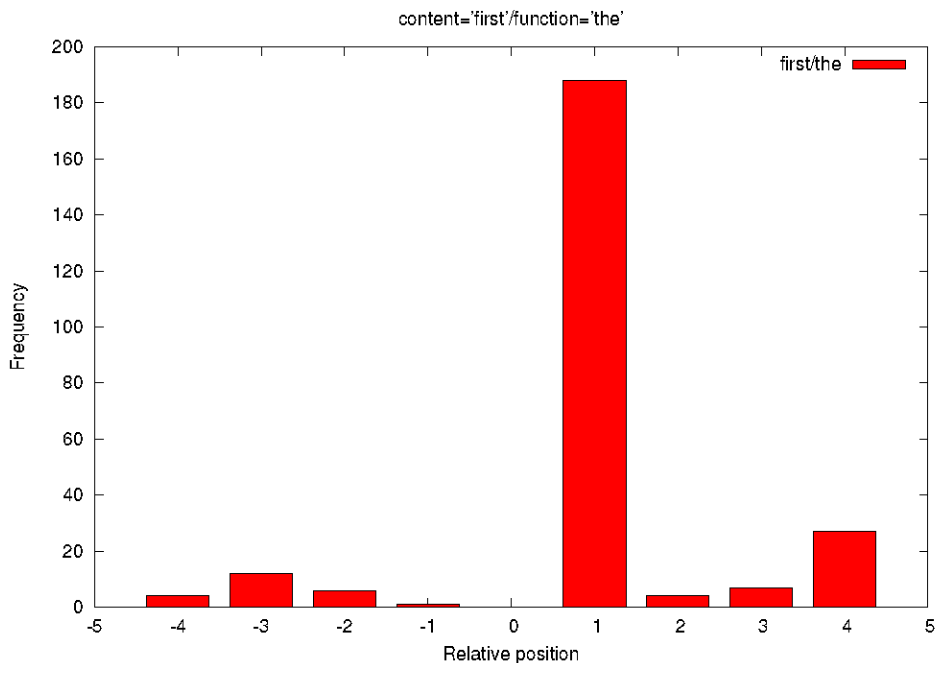


Figure 7: Graph showing how *first* is positioned relative to *the*

By treating each possible position of the target content word as a separate dimension, it is possible to represent the graph as a vector storing the frequency of occurrence for each dimension.

$$\begin{bmatrix} 4 \\ 12 \\ 6 \\ 1 \\ 0 \\ 188 \\ 4 \\ 7 \\ 27 \end{bmatrix}$$

An equivalent vector is needed for every function word in *F*, and the vectors concatenated to create a single vector that represents a profile of the content word, *w*, with respect to the set of function words, *F*. Once this has been applied to all of the words in *C*, a full set of vectors has been acquired, and these represent points in a high-dimensional vector space ready for the clustering phase.

5.2 Acquiring Function Words

The simplest way to acquire function words is to find an expert of the chosen language who can provide you with a list of them. This should be a finite, closed-class list. In practice, the list does not have to be perfect or complete, as clustering is an imperfect approximation. To acquire in a more automatic fashion, selecting the fifty most frequent words of a corpus will yield a reasonable list with adequate precision and recall. This is likely to return a few frequent content words, too.

Some function words are not that frequent, but are used consistently throughout a corpus. A fairer automatic method is to take a corpus and split it into a set of subcorpora. For each subcorpus, generate a word list, then the words that appear in all of the subcorpora are classed as function words. The exact proportions for the subcorpus size can depend on the corpus itself. For a million words, multi-genre corpus, having each subcorpus representing one percent of the overall corpus gives reasonable results (Roberts, 2006).

5.3 Clustering Algorithms

There are a range of clustering algorithms available. Hierarchical agglomerative algorithms are well-suited to word clustering. Each word in the vector space starts off as an individual one-object cluster. The algorithm evaluates each pair of points to find the closest, which are deemed the most similar, and merges them to form a single cluster. The way in which algorithms compute the distances between clusters is typically the differentiating factor (Everitt, 1993).

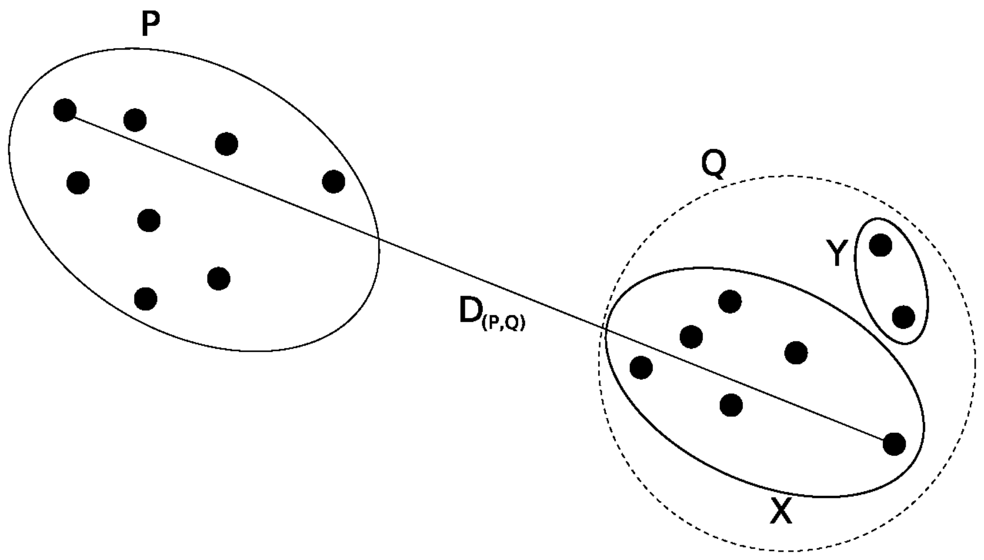


Figure 8: Diagram illustrating 'furthest neighbour' clustering algorithm

Figure 8 shows an example of the Complete Linkage algorithm. This is also known as the *furthest neighbour* method since its measure of distance between two clusters is to find the furthest two points within. Some of the more sophisticated, albeit, less simple to visualise, include Group Average and Ward's Method, which are good performers in this task (Zupan, 1982; Hughes and Atwell, 1994). The results return as a set of clusters, the size of which can vary. A nice feature of hierarchical methods is that you can profile the clustering process and generate a dendrogram that reveals how the cluster was formed, e.g., which terms were the most similar (see Figure 9).

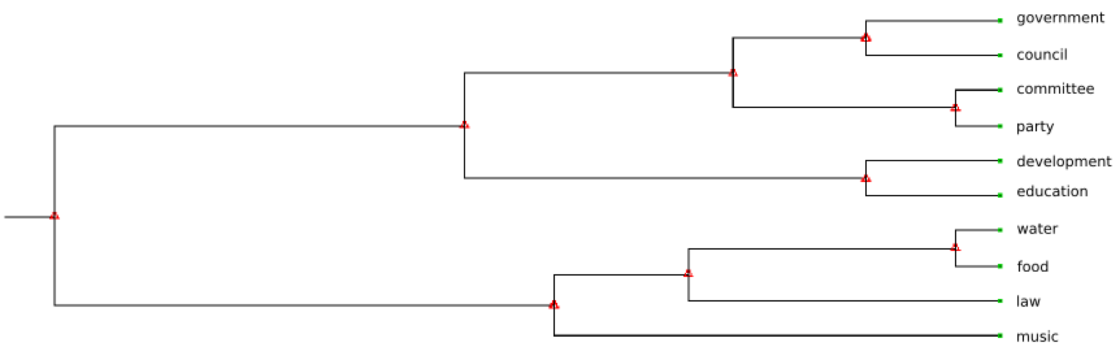
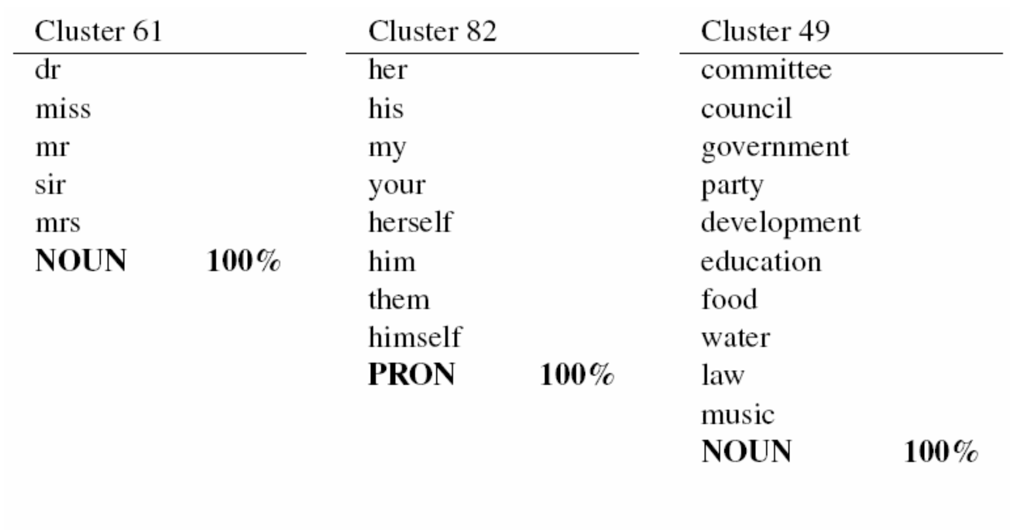


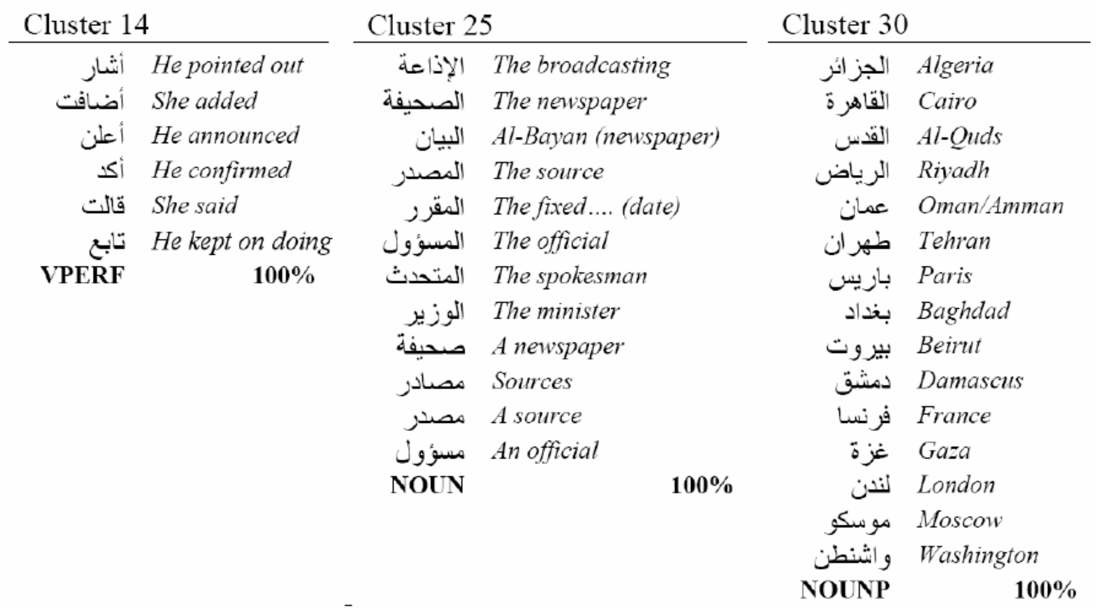
Figure 9: An example of a cluster visualised as a dendrogram

Even though clustering is an unsupervised approach, such algorithms have reported accuracies of 85+ percent in correctly grouping words into course-grained parts-of-speech (Roberts, 2002). Figures 10 and 11 show a few samples of the clustering output. A common drawback with clustering is that it can be prone to merging two well-defined clusters, (a set of nouns with a set of adjectives, for example), because at a given stage, they happened to be the 'closest'. The clustering algorithm has to know, as it were, when to stop, and this can be difficult for the program to judge automatically. If the algorithm stops too soon, the result is a large number of clusters each containing only a few words, which is unlikely to be very useful; and if the algorithm stops too late, there will be a small set of clusters each containing many words with not much (apparently) in common.



Cluster 61		Cluster 82		Cluster 49	
dr		her		committee	
miss		his		council	
mr		my		government	
sir		your		party	
mrs		herself		development	
NOUN	**100%**	him		education	
		them		food	
		himself		water	
		PRON	**100%**	law	
				music	
				NOUN	**100%**

Figure 10: Examples of clusters obtained from the LOB corpus



Cluster 14	
أشار	*He pointed out*
أضافت	*She added*
أعلن	*He announced*
أكد	*He confirmed*
قالت	*She said*
تابع	*He kept on doing*
VPERF	**100%**

Cluster 25	
الإذاعة	*The broadcasting*
الصحيفة	*The newspaper*
البيان	*Al-Bayan (newspaper)*
المصدر	*The source*
المقرر	*The fixed.... (date)*
المسؤول	*The official*
المتحدث	*The spokesman*
الوزير	*The minister*
صحيفة	*A newspaper*
مصادر	*Sources*
مصدر	*A source*
مسؤول	*An official*
NOUN	**100%**

Cluster 30	
الجزائر	*Algeria*
القاهرة	*Cairo*
القدس	*Al-Quds*
الرياض	*Riyadh*
عمان	*Oman/Amman*
طهران	*Tehran*
باريس	*Paris*
بغداد	*Baghdad*
بيروت	*Beirut*
دمشق	*Damascus*
فرنسا	*France*
غزة	*Gaza*
لندن	*London*
موسكو	*Moscow*
واشنطن	*Washington*
NOUNP	**100%**

Figure 11: Examples of clusters obtained from the Arabic Treebank corpus

6. Conclusion

This paper has summarised many of the issues regarding the difficulty of Arabic concordance. The *aConCorde* project attempts to resolve some of the core issues, for example, interactive searching that displays the Arabic text correctly and stem-based searching. However, at the time of writing, *aConCorde* is still limited in terms of its features compared to the more established products on the market, these include:

- Mark-up: *aConCorde* is generally ignorant of mark-up annotation within a corpus. Whilst it can successfully parse XML and HTML files, it essentially works by filtering out the tags to isolate the content. If a corpus was annotated with part-of-speech tags *aConCorde* would not recognise them and they would be ignored (even if they were encoded in XML), or treated as "words", as if part of the text itself. Heavily annotated corpora could therefore benefit from some pre-processing to strip away such mark-up.
- Full context: *aConCorde* does not have the functionality to allow the user to see the full context of a selected concordance item. The current display is to see the word within the sentence it was found, and that is the largest scope currently implemented.
- Arabic root/stem database is always exhaustive rather than reflecting only the tokens available in the corpus that is currently loaded. This could be a disadvantage in some circumstances as the user may have to browse through many stems that have no coverage.
- Clustering integration is at present crude and takes a long time to gather and display the data.
- Computational resources: *aConCorde* can be resource intensive with large datasets. Whilst the software utilises extremely scalable indexing technologies[8] similar to those found in modern databases (e.g., binary

[8] Courtesy of the Lucene indexing library: http://lucene.apache.org/

inverted indexes), loading corpora (10,000 words per seconds), or gathering the concordance for highly frequent words, takes time (retrieving 3,000 results could take a few seconds). However, once corpora are loaded, those files do not need to be reloaded and will be available immediately for all subsequent instantiations of the concordance software.

Naturally, the development of *aConCorde* will continue, and the limitations mentioned here are high priority issues that will be addressed in future issues. However, *aConCorde* has already made significant progress. The development of a simple user-interface with novel exploration features mean that it is well suited to a learning environment. With the foundation firmly laid, we can focus on other tasks in future, for example, improving and creating fresh approaches to intuitive morphological searching, or improving word similarity exploration through machine learning methods such as clustering. We also hope that the established concordance tools will realise the demand for products that cope with Arabic script, and adapt their software accordingly.

References

Al-Sulaiti, L. 2004. Designing and Developing a Corpus of Contemporary Arabic. Masters thesis, School of Computing, University of Leeds, UK.

Al-Sulaiti, L. and Atwell, E. 2006. 'The design of a Corpus of Contemporary Arabic', International Journal of Corpus Linguistics, 11 (2).

Aston, G. and Burnard, L. 1998. The BNC Handbook: Exploring the British National Corpus with SARA. Edinburgh: Edinburgh University Press.

Atwell, E. and Drakos, N. 1987. 'Pattern recognition applied to the acquisition of a grammatical classification system from unrestricted English text' in B. Maegaard (ed.) Proceedings of EACL: the Third Conference of European Chapter of the Association for Computational Linguistics. New Jersey: ACL.

Beesley, K. 2003. Xerox Arabic Morphological Analyzer Surface-Language (Unicode) Documentation. Xerox Research Centre Europe.

Boualem, M., Leisher, M. and Ogden, B. 1999. 'Concordancer for Arabic' in Arabic Translation and Localisation Symposium, Tunis, URL: http://crl.nmsu.edu/~mleisher/concord.pdf

Buckwalter, T. 2002. Buckwalter Arabic transliteration. URL: http://www.qamus.org/transliteration.htm

Burnard, L. 2004. 'BNC-Baby and Xaira' in Proceedings of the Sixth Teaching and Language Corpora conference, p. 84, Granada.

Burnard, L. and Dodd, T. 2003. 'Xara: an XML aware tool for corpus searching' in D. Archer, P. Rayson, A. Wilson and T. McEnery (eds.) Proceedings of the Corpus Linguistics 2003 Conference, pp. 142–44. University of Lancaster: UCREL.

Cobb, T., Greaves, C. and Horst, M. 2001. 'Can the rate of lexical acquisition from reading be increased? An experiment in reading French with a suite of online resources' in P. Raymond and C. Cornaire (eds.) Regards sur la didactique des langues secondes. Montréal: Éditions Logique.

de Roeck, A. 2002. 'Arabic for the absolute beginner', ELRA Newsletter, 7 (1).

Dodd, B. 1997. ‘Exploiting a corpus of written German for advanced language learning’ in A. Wichmann, S. Fligelstone, G. Knowles and T. McEnery (eds.) Teaching and Language Corpora, pp. 131–45. London: Longman.

Dodd, T. 1997. SARA: Technical Manual. URL: http://www.natcorp.ox.ac.uk/sara/TechMan/

Everitt, B. 1993. Cluster Analysis (third edition). London: Edward Arnold.

Finch, S. and Chater, N. 1992. ‘Bootstrapping syntactic categories’ in Proceedings of the 14th Annual Meeting of the Cognitive Science Society, pp. 820–25. Hillsdale, New Jersey.

Graddol, D. 1997. The Future of English? London: British Council.

Heddaya, A. 1985. Qalam: A convention for morphological Arabic-Latin-Arabic transliteration. URL: http://eserver.org/langs/qalam.txt

Hoogland, J. 2003. The Nijmegen Arabic/Dutch dictionary project—using the concordance program. URL: http://www.let.kun.nl/wba/Content2/1.4.6_Concordancing.htm

Hughes, J. 1994. Automatically Acquiring a Classification of Words. Ph.D. thesis, School of Computing, University of Leeds.

Hughes, J and Atwell, E. 1994. ‘The automated evaluation of inferred word classifications’ in A Cohn (ed.) Proceedings of ECAI94: 11th European Conference on Artificial Intelligence, pp. 535–40. Chichester: John Wiley.

Ibrahimov, O., Sethi, I. and Dimitrova, N. 2001. ‘Clustering of imperfect transcripts using a novel similarity measure’ in Proceedings of the SIGIR’01 Workshop on Information Retrieval Techniques for Speech Applications.

Johns, T. 1990. ‘From printout to handout: Grammar and vocabulary teaching in the context of data-driven learning’, Computer Assisted Language Learning, 10, pp. 14–34.

Jurafsky, D. and Martin, J. 2000. Speech and Language Processing: an Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. Upper Saddle River NJ: Prentice-Hall Pearson.

Khoja, S. 2003. APT: An Automatic Arabic Part-of-Speech Tagger. Ph.D. thesis, Computing Department, Lancaster University, UK.

Lawler, J. 2000. ‘Review of MonoConc Pro 2.0 concordancing software’, Linguist, 11 (1411).

Leech, G. and Fligelstone, S. 1992. ‘Computers and corpus analysis’ in C. Butler (ed.) Computers and Written Texts, pp. 115–40. Oxford: Blackwell.

Maamouri, M., Bies, A., Buckwalter, T. and Mekki, W. 2004. ‘The Penn Arabic Treebank: Building a large-scale annotated Arabic corpus’ in NEMLAR International Conference on Arabic Language Resources and Tools, Cairo, Egypt.

Ogden, B. and Bernick, P. 1996. ‘Oleda: User-centered Tipster technology for language instruction’ in Proceedings of the Tipster Pharse II 24 Month Workshop, VA, USA.

Roberts, A. 2002. Automatic acquisition of word classification using distributional analysis of content words with respect to function words. Technical report, School of Computing, University of Leeds.

Roberts, A. 2006. Effectiveness of Intersection Method for automatic acquisition of function words. Technical report, School of Computing, University of Leeds.

Roberts, A. and Atwell, E. 2005. 'aConCorde: Towards a proper concordance of Arabic' in P. Danielsson and M. Wagenmakers (eds.) Proceedings of the Corpus Linguistics 2005 Conference, University of Birmingham, UK.

Scott, M. 2004. WordSmith Tools 4.0. URL: http://www.lexically.net/downloads/version4/html/index.html

Sinclair, J.M. 1987. Looking up; an Account of the COBUILD Project. London: Collins ELT.

Zernik, U. 1991. 'Tagging word senses in corpus' in U. Zernik (ed.) Lexical Acquisition: Exploiting On-Line Resources to Build a Lexicon, pp. 91–112. Hillsdale, NJ: Lawrence Erlbaum.

Zupan, J. 1982. Clustering of Large Data Sets. Chichester: John Wiley and Sons.